



# UNITED STATES PATENT AND TRADEMARK OFFICE

*MN*  
UNITED STATES DEPARTMENT OF COMMERCE  
United States Patent and Trademark Office  
Address: COMMISSIONER FOR PATENTS  
P.O. Box 1450  
Alexandria, Virginia 22313-1450  
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/872,091	06/01/2001	Yasuteru Araya	14669	2464

23389 7590 06/25/2007  
SCULLY SCOTT MURPHY & PRESSER, PC  
400 GARDEN CITY PLAZA  
SUITE 300  
GARDEN CITY, NY 11530

EXAMINER

THANGAVELU, KANDASAMY

ART UNIT	PAPER NUMBER
----------	--------------

2123

MAIL DATE	DELIVERY MODE
-----------	---------------

06/25/2007

PAPER

**Please find below and/or attached an Office communication concerning this application or proceeding.**

The time period for reply, if any, is set in the attached communication.

<b>Office Action Summary</b>	<b>Application No.</b> 09/872,091	<b>Applicant(s)</b> ARAYA ET AL.	
	<b>Examiner</b> Kandasamy Thangavelu	<b>Art Unit</b> 2123	

**-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --**

**Period for Reply**

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

**Status**

- 1) ☒ Responsive to communication(s) filed on 01 May 2007.
- 2a) ☐ This action is **FINAL**.                      2b) ☒ This action is non-final.
- 3) ☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

**Disposition of Claims**

- 4) ☒ Claim(s) 11-26 is/are pending in the application.
- 4a) Of the above claim(s) \_\_\_\_\_ is/are withdrawn from consideration.
- 5) ☐ Claim(s) \_\_\_\_\_ is/are allowed.
- 6) ☒ Claim(s) 11-26 is/are rejected.
- 7) ☐ Claim(s) \_\_\_\_\_ is/are objected to.
- 8) ☐ Claim(s) \_\_\_\_\_ are subject to restriction and/or election requirement.

**Application Papers**

- 9) ☐ The specification is objected to by the Examiner.
- 10) ☒ The drawing(s) filed on 01 June 2001 is/are: a) ☒ accepted or b) ☐ objected to by the Examiner.  
     Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).  
     Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).
- 11) ☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

**Priority under 35 U.S.C. § 119**

- 12) ☒ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).
- a) ☒ All    b) ☐ Some \*    c) ☐ None of:
1. ☒ Certified copies of the priority documents have been received.
  2. ☐ Certified copies of the priority documents have been received in Application No. \_\_\_\_\_.
  3. ☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

\* See the attached detailed Office action for a list of the certified copies not received.

**Attachment(s)**

- |  |   |
|--|---|
| 1) <input checked="" type="checkbox"/> Notice of References Cited (PTO-892)                                | 4) <input type="checkbox"/> Interview Summary (PTO-413)<br>Paper No(s)/Mail Date. _____ |
| 2) <input type="checkbox"/> Notice of Draftsperson's Patent Drawing Review (PTO-948)                       | 5) <input type="checkbox"/> Notice of Informal Patent Application                       |
| 3) <input type="checkbox"/> Information Disclosure Statement(s) (PTO/SB/08)<br>Paper No(s)/Mail Date _____ | 6) <input type="checkbox"/> Other: _____  |

## **DETAILED ACTION**

1. This communication is in response to the Applicants' Amendment dated May 1, 2007. Claims 11 and 20 were amended. Claims 11-26 of the application are pending. This office action is made non-final.

### ***Claim Objections***

2. The following is a quotation of 37 C.F.R § 1.75 (d)(1):

The claim or claims must conform to the invention as set forth in the remainder of the specification and terms and phrases in the claims must find clear support or antecedent basis in the description so that the meaning of the terms in the claims may be ascertainable by reference to the description.

3. Claims 11, 14 and 20-22 are objected to because of the following informalities:

Claim 11, Lines 6-7, "by isolating elements of said source code representing hardware units and software units" appears to be incorrect and it appears that it should be "by isolating elements of said source code into elements representing hardware units and elements representing software units".

Claim 14, Lines 3-5, "by isolating in said source code new elements representing hardware units and new elements representing software units" appears to be incorrect and it appears that it should be "by isolating elements of said source code into new elements representing hardware units and new elements representing software units".

Art Unit: 2123

Claim 20, Line 6, "by isolating said source code in hardware and software elements" appears to be incorrect and it appears that it should be "by isolating said source code into hardware elements and software elements".

Claim 20, Lines 9-10, "the source code effecting said data transfer" appears to be incorrect and it appears that it should be "the source code effects said data transfer".

Claim 21, Lines 3-4, "by re-isolating said source code into hardware elements and software elements" appears to be incorrect and it appears that it should be "by re-isolating said source code in hardware and software elements".

Claim 22, Line 3, "isolation of the source code in hardware and software elements" appears to be incorrect and it appears that it should be "isolation of the source code into hardware elements and software elements".

Claims rejected but not specifically addressed are rejected, because of their dependence on rejected claims.

Appropriate corrections are required.

***Claim Rejections - 35 USC § 112***

4. The following is a quotation of the second paragraph of 35 U.S.C. 112:

The specification shall conclude with one or more claims particularly pointing out and distinctly claiming the subject matter which the applicant regards as his invention.

Art Unit: 2123

5. Claim 12, 13, 14 and 20 are rejected under 35 U.S.C. 112, second paragraph, as being indefinite for failing to particularly point out and distinctly claim the subject matter which applicant regards as the invention.

In claim 12, Line 2 "the evaluated data traffic" has insufficient antecedent basis.

In claim 13, Line 2 "the evaluated data traffic" has insufficient antecedent basis.

In claim 14, Line 2 "the step" has insufficient antecedent basis.

In claim 20, Line 3 "said architecture design" has insufficient antecedent basis.

In claim 20, Line 15 "the variable" has insufficient antecedent basis.

In claim 20, Line 18 "the architecture design level" has insufficient antecedent basis.

In claim 20, Line 22 "the bus traffic" has insufficient antecedent basis.

6. Claim 11-26 are rejected under 35 U.S.C. 112, second paragraph, as being indefinite for failing to particularly point out and distinctly claim the subject matter which applicant regards as the invention. This is because these claims use terms that are vague and indefinite.

6.1 Claim 11, Lines 18-19 state, "wherein said performance evaluation is used to modify an architecture design of an LSI". It is not clear where an architecture design is first created and what the applicant considers as an architecture design.

Art Unit: 2123

6.2 Claim 15, Lines 2-3 state, "isolation of the source code into elements representing hardware units and elements representing software units is optimized".

Claim 14 states, "isolating in said source code new elements representing hardware units and new elements representing software units". Why is the source code isolated into new elements in claim 14 and into elements in claim 15 and what is the difference between the two?

6.3 Claim 16, Lines 5-6 state, "repeating the foregoing steps until the source code is completely read in up to a last line of source code". The applicant has not said what the foregoing steps are, since "the steps" are not mentioned in claim 11 or 16 before. Does this mean all steps starting from structuring source code in claim 11 till performing said performance evaluation in claim 11 and modifying the source code in claim 16? If so, that is incorrect comparing that statement to Fig. 4 of the specification and its description.

6.4 Claim 20, Lines 16-17 state, "repeating the foregoing steps until the source code is completely read in and modified up to a last line of source code". Does this mean all steps starting from structuring a source code till modifying the source code? If so, that is incorrect comparing that statement to Fig. 4 of the specification and its description.

Art Unit: 2123

6.5 Claim 20, Line 23 states, "a given processing rate that is already known". What is this "a given processing rate"? Is it the processing rate of the bus or the processing rate of the processor of the LSI or the processing rate of the simulator processor?

6.6 Claim 20, Line 28 state, "wherein said performance evaluation is used to modify an architecture design of an LSI". It is not clear where an architecture design is first created and what the applicant considers as an architecture design.

Claims rejected but not specifically addressed are rejected because of their dependence on rejected claims.

### ***Claim Rejections - 35 USC § 103***

7. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

(a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains.

8. The factual inquiries set forth in *Graham v. John Deere Co.*, 383 U.S. 1, 148 USPQ 459 (1966), that are applied for establishing a background for determining obviousness under 35 U.S.C. 103(a) are summarized as follows:

1. Determining the scope and contents of the prior art.
2. Ascertaining the differences between the prior art and the claims at issue.

Art Unit: 2123

3. Resolving the level of ordinary skill in the pertinent art.
4. Considering objective evidence present in the application indicating obviousness or nonobviousness.

9. Claims 11-26 are rejected under 35 U.S.C. 103(a) as being unpatentable over **Tammemae et al.** ("AKKA: A tool for cosynthesis and prototyping", The Institution of Electrical Engineers, UK, 1996) in view of **Raimi et al.** (U.S. Patent 5,604,895), and further in view of **Chang et al.** (U.S. Patent 6,269,467).

9.1 **Tammemae et al.** teaches AKKA: A tool for cosynthesis and prototyping. Specifically, as per claim 11, **Tammemae et al.** teaches a method in a LSI design and development process (Page 1, Para 1, L1-2; Page 1, Para 2, L2-5), for evaluating an architecture design for an algorithm design by performing a performance evaluation of at least one bus at a high-level stage of the design and development process (Page 1, Para 2, L2-4; Page 1, Para 4, L2; Page 2, Para 4, L3); the method comprising:

structuring source code describing the algorithm design in a general purpose high-level programming language (Page 1, Para 2, L2-3; Page 1, Para 3, L1-3), by isolating elements of the source code representing hardware units and software units (Page 1, Para 2, L3; Page 1, Para 3, L7-8; Page 3, Fig. 1);

creating an evaluation function for counting data traffic that occurs on the at least one bus (Page 1, Para 4, L2; Page 1, Para 2, L3-4; Page 2, Para 4, L3: data transfer profiling; Page 2, Para 5, L1-3: during data transfer profiling for each variable access, a call to a counter function is made, which logs the access of the variable; before the target program exits, it calls a function



Art Unit: 2123

which writes the logged data to the file; Page 3, Fig. 1: compile for data transfer profiling; execute to get data transfer profiling information; Page 2, Para 2, L4), the bus being a part of the source code realizing the data traffic between the elements representing hardware units and software units (Page 1, Para 4, L2: we have added the data transfer profiling info; Page 1, Para 2, L3-4: partition the design into HW and SW components on the basis of execution time and time spent in communicating between HW and SW partitions; Page 1, Para 5, L5-6: 32-bit bus simplifies HW/SW co-simulation; Page 2, Para 2, L4; Page 2, Para 5, L1-3: during data transfer profiling for each variable access, a call to a counter function is made, which logs the access of the variable; before the target program exits, it calls a function which writes the logged data to the file; Page 3, Fig. 1: compile for data transfer profiling; execute to get data transfer profiling information; Co-verification between C/C++ Code and VHDL code: therefore this co-verification is done in source code, which means the communication bus is modeled in the source code and the profiling information is also added to the source code; Page 4, Para 2, Lines 1-2: for HW/SW co-simulation, set of C functions is provided which are implementing communication between SW and HW using IP sockets; therefore the bus in the co-simulation is part of the source code realizing data traffic between the elements representing hardware units and software units);

determining whether the source code was to be modified based on whether a line of source code represented writing data to variables that were defined in advance and were loaded onto the bus to be evaluated (Page 2, Para 5, L1-3);

modifying at least one element of the source code elements based on a result of an implementation of the evaluation function (Page 2, Para 5, L1-3: during data transfer profiling

Art Unit: 2123

for each variable access, a call to a counter function is made, which logs the access of the variable; before the target program exits, it calls a function which writes the logged data to the file; Page 2, Para 4, L3: data transfer profiling; Page 2, Para 2, L4: additional profiling functions inserted into an executable code);

performing the performance evaluation by simulating the modified source code elements and counting the data traffic on the bus (Page 2, Para 5, L1-3: during data transfer profiling for each variable access, a call to a counter function is made, which logs the access of the variable; before the target program exits, it calls a function which writes the logged data to the file; Page 3, Fig. 1: HW/SW compiling and co-verification; Page 4, Para 1, L1-3: find a solution that gives highest system speed up without increasing the system bus load; to get the information about bus load and minimize it; ).

**Tammemaie et al.** teaches that during data transfer profiling for each variable access, a call to a counter function is made, which logs the access of the variable (Page 2, Para 5, L1-3). **Tammemaie et al.** does not expressly teach sequentially reading in the source code line by line while effecting syntax analysis. **Raimi et al.** teaches sequentially reading in the source code line by line while effecting syntax analysis (Abstract, L5-10: the CPU parses the high level language description, to allow for generation of new code; CL2, L20-25: the method begins by parsing the original high level language description having at least one executable statement to obtain information; a high level language description, which includes new code generated in response to the information and the code of the original high level language description is generated; CL1, L26-32: profiling is the practice of surrounding the existing computer code of a high level

Art Unit: 2123

language program with monitor code; profiling is synonymous with instrumenting; CL1, L64-66: profiling is generation of additional code to check for the occurrence of various events). It would have been obvious to one of ordinary skill in the art at the time of Applicants' invention to modify the method of **Tammemae et al.** with the method of **Raimi et al.** that included sequentially reading in the source code line by line while effecting syntax analysis because that would allow generation of additional code to check for the occurrence of bus access events (CL1, L65-66).

**Tammemae et al.** and **Raimi et al.** do not expressly teach that said performance evaluation is used to modify an architecture design of an LSI. **Chang et al.** teaches that said performance evaluation is used to modify an architecture design of an LSI (CL27, L19-27: the designer constructs a high level diagram of the bus structure for the design; buses are designed to meet the required system performance while minimizing the interface logic; designers use this architecture to create a bus functional model to verify that the design operates as defined in the specification; CL51, L66 to CL52, L3: chip level verification comprises simulating the block functional model; any discrepancies are resolved by modifying one or more block functional models). It would have been obvious to one of ordinary skill in the art at the time of Applicants' invention to modify the method of **Tammemae et al.** and **Raimi et al.** with the method of **Chang et al.** that included said performance evaluation being used to modify an architecture design of an LSI because that would allow to meet the required system performance (CL27, L23-24); and as per **Tammemae et al.** to minimize the bus load (Page 4, Para 1, L2-3).

Art Unit: 2123

Per claim 12: **Tammemae et al.** teaches restructuring the source code based on the evaluated data traffic (Page 1, Para 2, L3-4; Page 1, Para 4, L1-2); and

performing the performance evaluation again by simulating the restructured source code again (Page 2, Para 5, L1-3; Page 3, Fig. 1; Page 4, Para 1).

Per claim 13: **Tammemae et al.** teaches that a bus traffic is calculated from the evaluated data traffic with respect to the processing rate of the bus (Page 2, Para 5, L1-3).

Per claim 14: **Tammemae et al.** teaches feeding back a result of the performance evaluation of the bus to the step of structuring the source code to improve the architecture design at a high-level design stage by isolating in the source code new elements representing hardware units and new elements representing software units (Page 1, Para 2, L3-4; Page 1, Para 4, L1-2; Page 2, Para 5, L1-3).

Per claim 15: **Tammemae et al.** teaches that in response to the bus traffic, isolation of the source code into elements representing hardware units and elements representing software units is optimized (Page 1, Para 2, L3-4; Page 1, Para 4, L1-2; Page 2, Para 5, L1-3).

9.2 As per claim 16, **Tammemae et al.**, **Raimi et al.** and **Chang et al.** teach the method of claim 11. **Tammemae et al.** teaches profiling the source code based on whether a line of source code represents writing data to variables that are defined in advance and are loaded onto the bus to be evaluated Page 2, Para 2, L4: additional profiling functions inserted into an executable

Art Unit: 2123

code; Page 2, Para 4, L3: data transfer profiling; Page 2, Para 5, L1-3: during data transfer profiling for each variable access, a call to a counter function is made, which logs the access of the variable; before the target program exits, it calls a function which writes the logged data to the file);

structuring the source code into elements representing at least one of the hardware units and the software units for use in the architecture design by compiling the source code (Page 1, Para 2, L3; Page 1, Para 3, L7-8; Page 3, Fig. 1);

calculating the data transfer rate on the bus by executing the compiled source code elements in a simulation program (Page 2, Para 5, L1-3: during data transfer profiling for each variable access, a call to a counter function is made; Page 3, Fig. 1; Page 4, Para 1, L2-3);

calculating bus traffic with regard to a given processing rate of the bus (Page 2, Para 5, L1-3: during data transfer profiling for each variable access, a call to a counter function is made); and

performing evaluation of the performance of the bus in response to the bus traffic (Page 2, Para 5, L1-3; Page 3, Fig. 1: HW/SW compiling and co-verification; Page 4, Para 1, L2-3).

**Tammemaie et al.** does not expressly teach upon determining that the source code is to be modified, modifying the source code by embedding the evaluation function one of immediately before or immediately after the line of source code in which the variable is written; and repeating the forgoing steps until the source code is completely read in up to a last line of source code. **Raimi et al.** teaches upon determining that the source code is to be modified, modifying the source code by embedding the evaluation function one of immediately before or

Art Unit: 2123

immediately after the line of source code in which the variable is written (CL1, L26-32: profiling is the practice of surrounding the existing computer code of a high level language program with monitor code; profiling is synonymous with instrumenting; CL1, L64-66: profiling is generation of additional code to check for the occurrence of various events; CL2, L20-25: the method begins by parsing the original high level language description having at least one executable statement to obtain information; a high level language description, which includes new code generated in response to the information and the code of the original high level language description is generated; and repeating the forgoing steps until the source code is completely read in up to a last line of source code (Abstract, L5-6; CL2, L20-22).

Per claim 17: **Tammemae et al.** teaches that the variables loaded onto the bus consist of  $n$  bits while the bus consists of  $m$  bit lines, where  $n$  and  $m$  are both integers, and  $n$  is a multiple of  $m$ , and the bus traffic for the processing rate is produced such that the number of times in effecting data transfer on the bus is multiplied by  $n/m$  and is then divided by the processing rate (Page 2, Para 5, L1-3). **Tammemae et al.** teaches that each variable access is counted using a counter and a log of the access of the variables is made. It is inherent that from the counting of the variable access to a bus, and the log of the variable accesses, it is possible to calculate the number of times the bus was used if the bus width was smaller than the variable length requiring multiple accesses to the bus for each variable loaded onto the bus.

Per claim 18: **Tammemae et al.** teaches that the general purpose high-level language is one of C language and C++ language (Page 1, Para 2, L2-3; Page 1, Para 3, L1-3).

Per claim 19: **Tammemae et al.** teaches that the evaluation function increments a counting value if a pre-defined variable is loaded onto the bus (Page 2, Para 5, L1).

9.3 As per claim 20, **Tammemae et al.** teaches a method in a LSI design and development process (Page 1, Para 1, L1-2; Page 1, Para 2, L2-5), for evaluating and facilitating an architectural design for an algorithm design by performing a performance evaluation of at least one bus at the architecture design being a high-level stage of the design and development process (Page 1, Para 2, L2-4; Page 1, Para 4, L2; Page 2, Para 4, L3); the method comprises the steps of:

structuring source code describing the algorithm design in a general purpose high-level programming language (Page 1, Para 2, L2-3; Page 1, Para 3, L1-3), by isolating the source code in hardware and software elements interconnected by the bus having a given bus configuration (Page 1, Para 2, L3-4; Page 1, Para 3, L7-8; Page 1, Para 4, L2; Page 1, Para 5, L5-6; Page 3, Fig. 1);

creating an evaluation function for evaluating data transfer that occurs on the bus (Page 1, Para 4, L2; Page 1, Para 2, L3-4; Page 2, Para 4, L3; Page 3, Fig. 1; Page 2, Para 2, L4), wherein the evaluation function counts and represents a number of times the source code effecting the data transfer on the bus (Page 2, Para 5, L1);

determining whether a line of source code represents writing data onto the bus to be evaluated (Page 2, Para 5, L1-3);

simulating the modified source code elements at the architecture design level and evaluating the data transfer on the bus (Page 2, Para 5, L1-3; Page 3, Fig. 1; Page 4, Para 1, L2-3);

compiling the structured source code elements (Page 1, Para 4, L1-2);

executing the compiled source code elements on a simulation platform (Page 2, Para 5, L1-3; Page 3, Fig. 1; Page 4, Para 1, L2-3);

calculating the bus traffic based on the number of times in effecting of data transfers according to the evaluation function and a given processing rate that is already known (Page 2, Para 5, L1-3);

performing the evaluation of the performance of the bus in response to the bus traffic being produced (Page 2, Para 5, L1-3; Page 3, Fig. 1; Page 4, Para 1, L2-3); and

wherein the method is carried out again in response to the performance evaluation if necessary by starting with changing the bus configuration and restructuring the source code (Page 2, Para 5, L1-3; Page 3, Fig. 1; Page 4, Para 1).

**Tammemae et al.** does not expressly teach sequentially reading in the source code; upon the determination, modifying the source code by embedding the evaluation function just before or after the line of source code in which the variable is written; and repeating the forgoing steps until the source code is completely read in and modified up to a last line of source code. **Raimi et al.** teaches sequentially reading in the source code (Abstract, L5-10; CL2, L20-25); upon the determination, modifying the source code by embedding the evaluation function just before or after the line of source code in which the variable is written (CL1, L26-32; CL1, L64-66; CL2,



Art Unit: 2123

L23-29); and repeating the forgoing steps until the source code is completely read in and modified up to a last line of source code (Abstract, L5-6; CL2, L20-22).

**Tammemae et al.** does not expressly teach that said performance evaluation is used to modify an architecture design of an LSI. **Chang et al.** teaches that said performance evaluation is used to modify an architecture design of an LSI (CL27, L19-27: the designer constructs a high level diagram of the bus structure for the design; buses are designed to meet the required system performance while minimizing the interface logic; designers use this architecture to create a bus functional model to verify that the design operates as defined in the specification; CL51, L66 to CL52, L3: chip level verification comprises simulating the block functional model; any discrepancies are resolved by modifying one or more block functional models).

Per claim 21: **Tammemae et al.** teaches feeding back a result of the performance evaluation of the bus to the step of structuring the source code to improve the architecture design at a high-level design stage by re-isolating the source code in hardware and software elements (Page 1, Para 2, L3-4; Page 1, Para 4, L1-2; Page 2, Para 5, L1-3).

9.4 As per claim 22, **Tammemae et al.**, **Raimi et al.** and **Chang et al.** teach the method of claim 20. **Tammemae et al.** teaches that the bus configuration comprises the number of bit lines of the bus (Page 2, Para 5, L1-3); and isolation of the source code in hardware and software elements is optimized (Page 1, Para 2, L3; Page 1, Para 3, L7-8; Page 3, Fig. 1).

**Tammemae et al.** and **Raimi et al.** do not expressly teach in response to the bus traffic changing the number of bit lines of the bus. **Chang et al.** teaches in response to the bus traffic changing the number of bit lines of the bus (CL27, L18-27; CL29, L29-35; CL33, L53-59; CL39, L1-7).

Per claim 23: **Tammemae et al.** does not expressly teach after creating the evaluation function, sequentially reading in the source code line by line while effecting syntax analysis. **Raimi et al.** teaches after creating the evaluation function, sequentially reading in the source code line by line while effecting syntax analysis (Abstract, L5-10; CL2, L20-252).

Per claim 24: **Tammemae et al.** teaches that the variables loaded onto the bus consist of  $n$  bits while the bus consists of  $m$  bit lines, where  $n$  and  $m$  are both integers, and  $n \geq m$ , and the bus traffic for the processing rate is produced such that the number of times in effecting data transfer on the bus is multiplied by  $n/m$  and is then divided by the processing rate (Page 2, Para 5, L1-3). **Tammemae et al.** teaches that each variable access is counted using a counter and a log of the access of the variables is made. It is inherent that from the counting of the variable access to a bus, and the log of the variable accesses, it is possible to calculate the number of times the bus was used if the bus width was smaller than the variable length requiring multiple accesses to the bus for each variable loaded onto the bus.

Per claim 25: **Tammemae et al.** teaches that the general purpose high-level language is one of C language and C++ language (Page 1, Para 2, L2-3; Page 1, Para 3, L1-3).

Per claim 26: **Tammemae et al.** teaches that the evaluation function is to increment a counting value if a pre-defined variable is loaded onto the bus (Page 2, Para 5, L1).

### ***Response to Arguments***

10. Applicants' arguments filed on May 1, 2007 with respect to 35 USC 103 (a) rejections are not persuasive. In addition, Applicants' amendments to claims 11 and 20 have necessitated using new 103 (a) rejections for those claims. Additional claim rejections under 35 USC 112 second Paragraph are included in this Office Action.

10.1 As per the applicants' argument that "Tammemae does not disclose or suggest the bus being modeled between the hardware and software elements", the Examiner respectfully disagrees.

**Tammemae et al.** teaches the bus being modeled between the hardware and software elements (Page 1, Para 4, L2: we have added the data transfer profiling info; Page 1, Para 2, L3-4: partition the design into HW and SW components on the basis of execution time and time spent in communicating between HW and SW partitions; Page 1, Para 5, L5-6: 32-bit bus simplifies HW/SW co-simulation; Page 2, Para 2, L4; Page 2, Para 5, L1-3: during data transfer profiling for each variable access, a call to a counter function is made, which logs the access of the variable; before the target program exits, it calls a function which writes the logged data to

the file; Page 3, Fig. 1: compile for data transfer profiling; execute to get data transfer profiling information; Co-verification between C/C++ Code and VHDL code: therefore this co-verification is done in source code, which means the communication bus is modeled in the source code and the profiling information is also added to the source code; Page 4, Para 2, Lines 1-2: for HW/SW co-simulation, set of C functions is provided which are implementing communication between SW and HW using IP sockets; therefore the bus in the co-simulation is part of the source code realizing data traffic between the elements representing hardware units and software units).

10.2 As per the applicants' argument that "Tammemae does not teach or suggest determining whether the source code is to be modified based on whether a line of source code represents writing data to variables that are defined in advance and are loaded onto the bus to be evaluated; because Raimi does not teach or suggest evaluating the performance of a bus which is part of source code, he does not teach determining whether the source code is to be modified based on whether a line of source code represents writing data to variables that are defined in advance and are loaded onto the bus to be evaluated", the Examiner respectfully disagrees.

**Tammemae et al.** teaches determining whether the source code was to be modified based on whether a line of source code represented writing data to variables that were defined in advance and were loaded onto the bus to be evaluated (Page 2, Para 5, L1-3: during data transfer profiling for each variable access, a call to a counter function is made, which logs the access of the variable; before the target program exits, it calls a function which writes the logged data to

Art Unit: 2123

the file; Page 2, Para 4, L3: data transfer profiling; Page 2, Para 2, L4: additional profiling functions inserted into an executable code).

### ***Conclusion***

11. The prior art made of record and not relied upon is considered pertinent to the applicant's disclosure.

The following patents are cited to further show the state of the art with respect to hardware software co-simulation using software models of hardware and bus interface models in software.

1. Klein et al., "Optimizing hardware and software co-verification system", U.S. Patent 6,212,489, April 2001 (filing date June 1998).


12. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Dr. Kandasamy Thangavelu whose telephone number is 571-272-3717. The examiner can normally be reached on Monday through Friday from 8:00 AM to 5:30 PM.

Art Unit: 2123

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Paul Rodriguez, can be reached on 571-272-3753. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Any inquiry of a general nature or relating to the status of this application or proceeding should be directed to TC 2100 Group receptionist: 571-272-2100.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see <http://pair-direct.uspto.gov>. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

  
K. Thangavelu  
Art Unit 2123  
June 20, 2007